UNIT - 5:

1.Consider two cloud service systems: Google File System and Amazon S3. Explain how they achieve their design goals to secure data integrity and to maintain data consistency while facing the problems of hardware failure, especially concurrent hardware failures.

Google File System (GFS) and Amazon Simple Storage Service (S3) are two popular cloud storage systems that aim to ensure data integrity and consistency in the presence of hardware failures, including concurrent failures. Here's how they achieve these design goals:

Google File System (GFS):

1. **Data integrity:** GFS stores data in large chunks (typically 64 MB or 128 MB) across multiple DataNodes. Each chunk is replicated on multiple machines for reliability. When a client writes data, GFS generates a checksum for each chunk and stores it along with the data. Upon reading the data, GFS recomputes the checksum and compares it with the stored checksum to ensure data integrity.
2. **Data consistency:** GFS uses a master node, called the Master, to manage the metadata of the file system, such as the mapping of files to chunks and the locations of the chunks. The Master also handles client requests for file operations. GFS uses a lease mechanism to ensure consistency. When a client obtains a lease from the Master for a file or a chunk, it has exclusive access to that file or chunk for a certain period. This mechanism prevents conflicts and ensures consistency.
3. **Handling hardware failures:** GFS is designed to handle hardware failures gracefully. When a DataNode fails, the Master detects the failure and automatically re-replicates the lost chunks on other DataNodes. GFS also has a mechanism to handle concurrent failures. If multiple DataNodes fail before the Master can recover the lost chunks, the system can still maintain data consistency.

Amazon Simple Storage Service (S3):

1. **Data integrity:** S3 stores data with multiple copies for durability. S3 calculates and verifies checksums for each object during upload, transfer, and retrieval. S3 also supports data versioning, allowing users

to preserve, retrieve, and restore every version of every object in their bucket.

2. **Data consistency:** S3 provides read-after-write consistency for PUT and DELETE requests on new objects in your bucket in all regions. For overwrite PUT and DELETE requests, S3 provides eventual consistency. This means that after a successful overwrite PUT or DELETE request, subsequent GET requests may return the prior version of the object or a 404 (Object Not Found) error for a short time.
3. **Handling hardware failures:** S3 is designed to handle hardware failures by automatically replicating data across multiple availability zones within a region. This redundancy ensures that data remains accessible even if one or more storage devices fail. S3 also uses erasure coding, a technique that encodes data across multiple devices, to protect against concurrent hardware failures.

In summary, both GFS and S3 employ various strategies to ensure data integrity, consistency, and resilience against hardware failures, including concurrent failures. GFS uses chunk replication, checksums, and a lease mechanism, while S3 relies on data redundancy, checksums, versioning, and erasure coding.

2.Security is a critical aspect of data management. Discuss the security features and mechanisms in HDFS, including authentication, authorization, and encryption. Explain how HDFS addresses potential security threats and safeguards data at rest and in transit. Consider scenarios where fine -grained access control is essential and how HDFS provides mechanisms for access restriction.

Hadoop Distributed File System (HDFS) has several security features to ensure the confidentiality, integrity, and availability of data. These features include authentication, authorization, and encryption for data at rest and in transit.

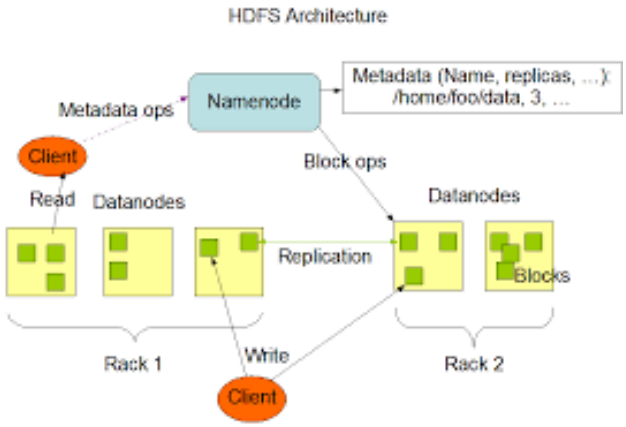
Authentication: HDFS uses Kerberos for authentication, which is a network authentication protocol that prevents unauthorized access to the system. With Kerberos, users and services must authenticate themselves before accessing HDFS resources.

Authorization: Access Control Lists (ACLs) and Apache Ranger provide fine-grained access control in HDFS. ACLs can be used to define permissions for individual users or groups on a per-file or per-directory basis.

Encryption: HDFS supports encryption for data at rest and in transit. Data encryption can be enabled through the HDFS encryption zone feature, which automatically encrypts files and directories. Data encryption can also be enforced during transit using HTTPS or TLS.

Access restriction: HDFS offers mechanisms for access restriction through ACLs, as mentioned earlier. Additionally, HDFS supports secure deletion of files, which ensures that deleted files are overwritten and cannot be recovered.

3.Predictive analytics helps healthcare organizations to anticipate change - so that you can plan and carry out strategies that improve results. By applying predictive analytics solutions to data you already have, your organization can uncover unexpected patterns and associations and develop models to improve clinical and operational performance. Explicate the Hadoop core components with suitable diagrams and justify the use, for the above scenario.



Hadoop Distributed File System (HDFS):

- Stores huge amounts of healthcare data securely across many computers.
- Healthcare organizations have lots of data like patient records and medical images. HDFS lets them store and access this data easily.

MapReduce:

- Quickly processes large datasets by splitting them into smaller parts and running them on multiple computers.
- Healthcare organizations can analyze massive amounts of data faster to find patterns and make better decisions.

YARN (Yet Another Resource Negotiator):

- Manages resources like memory and processing power so tasks can run smoothly on a cluster of computers.
- It ensures that healthcare analytics jobs run efficiently without any computer getting overloaded.

Hadoop Common:

- Provides tools and libraries needed for Hadoop to work properly.
- It makes it easier for healthcare organizations to process and analyze different types of data without needing separate tools.

Justification for Use in Healthcare Predictive Analytics:

- Scalability
- Performance
- Cost-effectiveness
- Data Variety

6.A cloud provider offers two types of instances for running MapReduce jobs: Type A costs $0.10 per hour and has 4 CPU cores, while Type B costs $0.15 per hour and has 8 CPU cores. Determine the most cost -effective instance type for a job that requires 10 hours of computation time.

Type A:

- Hourly rate: $0.107
- Number of CPU cores: 4
- Computation time: 10 hours

Total Cost for Type A = Hourly Rate * Number of Cores * Computation Time
= $0.10 * 4 * 10
= $4.00

Type B:

- Hourly rate: $0.15
- Number of CPU cores: 8
- Computation time: 10 hours

Total Cost for Type B = Hourly Rate * Number of Cores * Computation Time
= $0.15 * 8 * 10
= $12.00

Conclusion: For the given scenario with a computation time of 10 hours, Type A is the most cost-effective instance type as it has a total cost of $4.00, while Type B has a higher total cost of $12.00.

7.Suppose you have a dataset of 1000 records distributed across 4 nodes in a MapReduce cluster. Calculate the number of records processed by each mapper if the data is partitioned equally among the nodes.

Number of Records per Mapper = Total Records / Number of Nodes
In this case:
Number of Records per Mapper = 1000 / 4 = 250
Conclusion: Therefore, each mapper in the MapReduce cluster will process 250 records. This assumes an equal distribution of data across the nodes, providing a balanced workload for each mapper in the cluster.

8.In a MapReduce job, each mapper produces 50 intermediate key -value pairs on average, which need to be shuffled and sorted before being passed to reducers. If there are 20 mappers in total, calculate the total number of intermediate key - value pairs shuffled across the network.

Total Shuffled Pairs = Average Pairs per Mapper * Total Mappers
In this case:
Total Shuffled Pairs = 50 * 20 = 1000
Conclusion: Therefore, the total number of intermediate key-value pairs shuffled across the network in the MapReduce job is 1000.

9)A company needs to transfer 1 TB of data from its on-premises data center to a cloud storage service. The cloud provider charges $0.10 per GB for data transfer. Calculate the total cost of transferring the data to the cloud, expressed in USD

To calculate the total cost of transferring 1 TB of data to the cloud, you can use the following formula:
$$\text{Total Cost} = \text{Data Size} \times \text{Cost per GB}$$
Given that the data size is 1 TB and the cost per GB is $0.10, first convert the data size to gigabytes (1 TB = 1024 GB) and then plug the values into the
Formula:
$$\text{Total Cost} = 1024\ \text{GB} \times \$0.10/\text{GB}$$
$$\text{Total Cost} = \$102.40$$
So, the total cost of transferring 1 TB of data to the cloud is $102.40.

10)A Hadoop cluster consists of 20 nodes, each with 32 CPU cores and 64 GB of RAM. If a MapReduce job utilizes 90% of the CPU cores and 70% of the RAM on each node, calculate the total CPU core and RAM utilization across the cluster.

Given:

- Number of nodes in the cluster: 20
- CPU cores per node: 32
- Total RAM per node: 64 GB
- CPU core utilization: 90%
- RAM utilization: 70%

To calculate the total CPU core and RAM utilization across the Hadoop cluster, you can use the following formulas:

Total CPU Core Utilization:

Total CPU Core Utilization = Number of Nodes x Percentage of CPU Cores Utilized

Total CPU Core Utilization = 20 x 90% = 18 cores

Total RAM Utilization:

Total RAM Utilization = Number of Nodes x Percentage of RAM Utilized x Total RAM per Node

Total RAM Utilization = 20 x 70% x 64GB = 896 GB

So, the total CPU core utilization across the Hadoop cluster is 18 cores, and the total RAM utilization is 896 GB.